

The Bioinformatics Lab SS2011

May 23, 2011

Virtualization with KVM/QEMU

Report: Veit Hoehn (hoehn@cip.ifi.lmu.de)

1. Introduction

A virtual machine is a machine, which doesn't consist of hardware like the physical machines e.g. Desktop PCs or Laptops, but of software. Of course this software has still to run on a "real" machine. But here is the first advantage of a virtual machine. It is possible to run several virtual machines on one physical machine. The number of virtual machines is only limited by the hardware resources of the physical machine. Another important point for virtualization is that it is possible to have different architectures and different operating systems running on the same computer. If you now think of that the real machine can be a server standing somewhere and everybody can connect to it and then use his one virtual machine, you get an impression of the high variability and vast range of possibilities which are opened by the concept of virtualization. Just to give an example, when giving everyone in a lab his own virtual machine instead of all sharing one operating system with user management, everyone can have root rights and play around, install things without endanger the system stability. So if someone is accidentally crashing the system, he only crashes his system, not the whole system.

Today's task in the bioinformatics lab course is to set up a virtual machine on a server. The server is running on Linux and the virtual machine should be a Debian stable, like in the first exercise of the course. Second task is to install and manage two ways of accessing the virtual machine from any other machine. One way would be via X-forwarding, the other way involves a VNC server-client connection. The architecture of the real and virtual machines is shown in *Figure 1*.

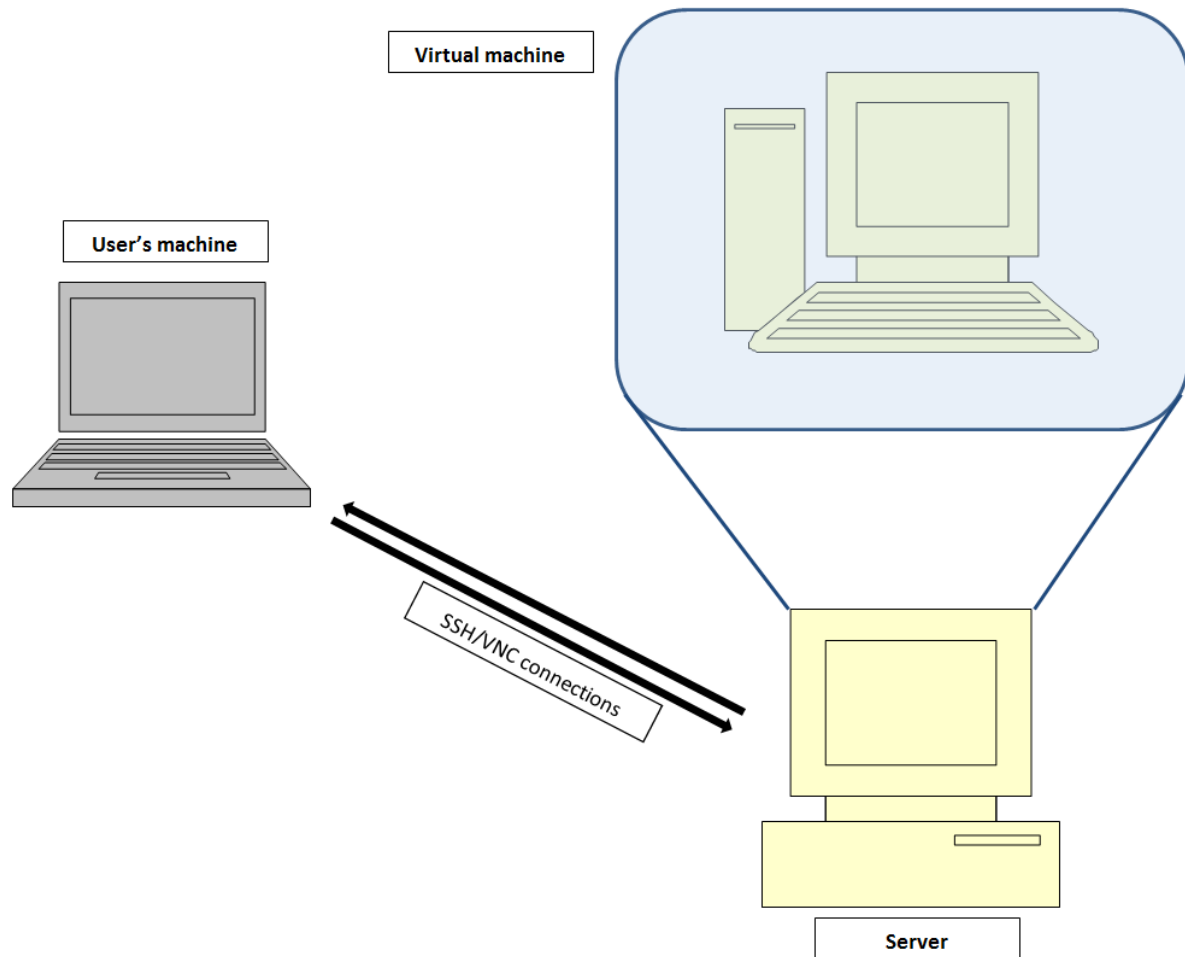


Figure 1: Basic layout of the virtual and real machines involved in today's task. On the Server and the User's machine Linux systems are running. KVM/QEMU is already installed on the Server.

2. Creating the virtual machine

First step of all is to connect to the Server via SSH. This simple task is done by the following command. With the “-X” option you enable forwarding of the X-window. This is necessary to see the installation menu of Debian.

```
ssh -X hoehn@il2r-studfilesrv.informatik.tu-muenchen.de
```

Command 1: Connecting to the Server via SSH.

On the server there is virtualization software installed, in this case it is KVM and QEMU. First task is therefore to create the virtual machine, which should be in our case a Debian stable system. Of course any other operating system would be possible, too.

First step is to create a virtual machine image. Before doing this a few thoughts about size and format should be done. Size in our case is limited to 2 gigabyte. Format should be the qcow2 format. This QEMU format is very versatile especially for small images like in our case.

```
kvm-img create -f qcow2 image.qcow2 2G
```

Command 2: Creating the virtual machine image with kvm-img.

We use the “kvm-img” command to create the image “image.qcow2” with 2 gigabyte size, specified by the “2G” at the end. With “kvm-img” there are lots of other configurations possible like compression and encryption, but for our example the simple way should do the job.

Next step is to boot the image and install the Debian stable on it. For this purpose we need an image of the Debian which can be retrieved from the Debian website. Our image of choice is the “amd64 businesscard iso”.

```
wget http://cdimage.debian.org/debian-cd/6.0.1a/amd64/iso-cd/debian-6.0.1a-amd64-businesscard.iso
```

Command 3: Getting the Debian installation image from the Debian website with wget.

Now that we have the installation source we can start the virtual machine. The starting of the virtual machine is not that easy because a lot of parameters are possible. In this example we only focus on the essential necessary commands to get the system up. Besides these commands there are a lot of commands involving for example enhancements in speed.

```
kvm -hda image.qcow2 -cdrom debian-6.0.1a-amd64-businesscard.iso -boot d -m 512 -net nic,model=virtio -net vde,sock=/var/run/kvm0.ctl
```

Command 4: Starting the virtual machine for the installation.

Command 4 shows how to start the virtual machine for installation. The first option “-hda” is determining the virtual machine image on which the Debian should be installed. The “-cdrom” option is specifying the CD-ROM in our case the Debian installation image. The “-boot d” tells the virtual machine to boot from CD-ROM first. The “-m 512” is limiting the memory used by the virtual machine to 512 megabytes. To enhance speed issues “-net nic,model=virtio” is used. This is changing the VLAN model to “virtio”. The “-net vde” command is configuring the virtual distributed ethernet, in this case determining the socket path. This is necessary for using the static IP addresses each machine should have. The installation might also work without the specified vde socket.

3. Installation of the Debian stable system

The installation is pretty similar to the one in the first lecture of the bioinformatics lab; there are only a few differences to consider. First one is due to the limited hard disc space (only 2 gigabytes). So we disclaim on a SWAP and a boot partition and install everything on one partition. Important here is to enable the bootable flag. Next thing, also due to the small disc space available, is not to install the graphical desktop environment.

Last differences to the first lecture, this time the GRUB bootloader has to be installed in the master boot record. Obviously, because the virtual machine is acting as a real machine, therefore needs a bootloader.

4. Configuration of the Debian stable system

To configure the newly installed Debian, we first have to boot it. After installation the virtual machine is rebooting, but because of the “-boot b” flag not the new Debian. So we first have to shut down the virtual machine. The QEMU command for this is “*system_powerdown*”. We have to enter this command in the hypervisor’s monitor, which is available after pressing [CTRL]+[ALT]+[2]. ([CTRL]+[ALT]+[1] to switch back)

Back in the Server shell, we now have to restart the virtual machine with following command.

```
kvm -hda image.qcow2 -m 512 -net nic,model=virtio -net  
vde,sock=/var/run/kvm0ctl
```

Command 5: Restarting the virtual machine with the newly installed Debian.

As you can see in *Command 5*, it is the same command like when starting the virtual machine for the first time, only without the “-cdrom” and the “-boot” option. So this time the virtual machine is booting the newly installed Debian.

Next step is to configure the users. In my case I have to create a user “hoehn” with the user id “1006”. The user “hoehn” was already created in the installation, but it has still the user id “1000”. So to change this we have to edit the **/etc/passwd** and the **/etc/shadow** file, see *Command 6*. You must be root to do this.

```
vi /etc/passwd /etc/shadow -o
```

Command 6: Opening the /etc/passwd and the /etc/shadow file to change the user id.

In both files you have to change the user id in the line “*hoehn:x:1000:1000...*” to “*hoehn:x:1006:1006...*”. This will also change the primary group id to “1006”. To create the group id “1006” it is also necessary to edit the **/etc/group** and the **/etc/gshadow** files. Here we have to search for “hoehn” and change the id once again from “1000” to “1006”.

Last step in the user and group configuration is to change the owner ship of the home directory to the new user id. For this we use Command 7.

```
chown -R hoehn: /home/hoehn
```

Command 7: Changing the ownership of all files in the home directory to the new id.

Last step of the configuration is to configure the static IP address for “eth0”. This is done by editing the **/etc/network/interface** file like described in *Table 1*. Here root rights are also necessary.

Old /etc/network/interface	New /etc/network/interface
iface eth0 inet DHCP	iface eth0 inet static address 192.168.16.6 netmask 255.255.255.0 gateway 192.168.16.1

Table 1: Old and new **/etc/network/interface file. New one contains the static IP address configuration.**

After this the network has to be restarted with following command.

```
/etc/init.d/networking restart
```

Command 8: Restart the network.

At this point, some additional software can be installed for more a convenient usage, but basically all the necessary work is done.

5. Accessing the virtual machine from outside

First way is via SSH including X-forwarding. This is the standard way and was also used for installation. So back on the Server shell we create a shell script which starts the virtual machine by opening a window which is then forwarded to our client computer via SSH X-forwarding. This very simple script is doing the same start up command like before and is showed below.

```
#!/bin/sh  
kvm -hda image.qcow2 -m 512 -net,model=virtio, -  
net,vde,sock=/var/run/kvm0ctl
```

Content of the first shell script: Starting the virtual machine with the default SDL display.

Second way to interact with the virtual machine is via VNC. The KVM hypervisor has a build in VNC server; therefore we just have to modify the starting command a little. To control the virtual machine we need to set up the monitor of the hypervisor. In our case we will use a unix socket. Also we do not want the virtual machine blocking the shell or crashing when logging out, so we have to start it in the background with “nohup”. The starting command now is like in the following shell script.

```
#!/bin/sh
nohup kvm -hda image.qcow2 -m 512 -net,model=virtio, -
net,vde,sock=/var/run/kvm0ctl -chardev
socket,id=monitor,path=tbl2011.monitor,server,nowait -mon
chardev=monitor,mode=readline -vnc none >/dev/null 2>&1 &
```

Content of the second shell script: Starting the virtual machine with nohup in the background and VNC server

The first new option in the start line of the virtual machine is to create a socket with id “monitor” in the file “tbl2011.monitor”. With the “server” in the “chardev” option it is specified that this socket is a listening one, the “nowait” tells that QEMU should not block waiting for a client to connect to the listening socket. The monitor is redirected to the socket with the “-mon” option in the “readline” mode. With “-vnc none” the VNC mode is enabled but no port is specified. Finally the standard output and error are redirected to **dev/null** to avoid that the nohup log files are getting too big. The “&” at the end is starting the command in the background; the nohup at the beginning is preventing the virtual machine from quitting when the user logs out from the server.

So after starting the virtual machine with the second shell script in the VNC mode, the next step is to specify a port to which a VNC client could connect. To do this we have to go the monitor of the hypervisor, which is specified as “tbl2011.monitor”. To do this we use *Command 9*.

```
nc -U tbl2011.monitor
```

Command 9: Connect to the hypervisor.

In the hypervisor monitor we can determine the VNC port with the command “*change vnc :20*” the last number plus 5900 is specifying the port. Of course the specified port must be free; only one virtual machine can use it.

So the Server and the virtual machine are both ready for use. In order to connect to the virtual machine on our server, the client machine needs a VNC viewer. Then because of firewall issues

on the server it is necessary to forward the specified port of the VNC server, in our case 5920 to the local machine. This can be done by following command on the local machine.

```
ssh -L 5920:127.0.0.1:5920 hoehn@i12r-studfilesrv.informatik.tu-  
muenchen.de
```

Command 10: Forwarding the VNC port from the server to the local client machine.

The “-L” option is forwarding the port 5920, the VNC port, from the ssh-server (which is 127.0.0.1 in the upper command) to the local client machine (also 127.0.0.1 but not the one in the upper command) also to the port 5920, see Figure 2. Now we have the port on the local machine and we can access it with the VNC viewer on the address 127.0.0.1:5920 and enjoy working on the virtual machine from everywhere.

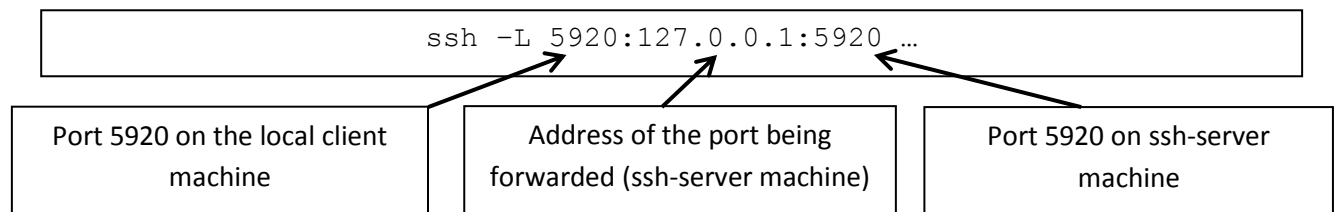


Figure 2: Scheme of the port forwarding command (ssh). Port 5920 from the server machine (127.0.0.1 for the ssh-server) is being forwarded to port 5920 on the local client machine. The port 5920 on our local client machine is therefore connected to the port 5920 on the server machine.